

UNITED STATES PATENT APPLICATION

FOR

MECHANISM FOR ENABLING FULL DATA BUS UTILIZATION
WITHOUT INCREASING DATA GRANULARITY

INVENTOR:

BILLY GARRETT, JR.

PREPARED BY:

HICKMAN PALERMO TRUONG & BECKER, LLP
1600 WILLOW STREET
SAN JOSE, CALIFORNIA 95125-5106
(408) 414-1080

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number EL624353176US

Date of Deposit 4-17-01

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Box Patent Application, Commissioner of Patents, Washington, D.C. 20231.

Tirena Say

(Typed or printed name of person mailing paper or fee)

Tirena Say
(Signature of person mailing paper or fee)

**MECHANISM FOR ENABLING FULL DATA BUS UTILIZATION
WITHOUT INCREASING DATA GRANULARITY**

Inventor(s): Billy Garrett, Jr.

5 Field of the Invention

 This invention relates generally to storage technology, and more particularly to a mechanism for enabling full data bus utilization in a memory without increasing data granularity.

10 Background

 Dynamic random access memories (DRAM's) are used as main memory in many of today's computer systems. One of the factors that have led to the popularity of the DRAM has been the DRAM's simple cell structure. Because each DRAM storage cell consists of just a single capacitor, it is possible to pack a very large number of storage
15 cells into a very small amount of chip space. Consequently, with DRAM technology, it is possible to manufacture very high-density, low cost memories.

 With reference to Fig. 1, there is shown a functional diagram of a typical DRAM 100. As shown in Fig. 1, a DRAM 100 comprises a plurality of memory cells 102 arranged in a plurality of rows and columns. Each row of memory cells 102 is coupled to
20 one of the wordlines 104 of the DRAM 100, and each column of memory cells 102 is coupled to one of the bitlines 106. By specifying a wordline 104 and a bitline 106, a particular memory cell 102 can be accessed.

To enable access to the various memory cells 102, there is provided a row decoder 112 and a column decoder 114. The row decoder 112 receives a row address on a set of address lines 116, and a row address strobe (RAS) signal on a control line 118, and in response to these signals, the row decoder 112 decodes the row address to select one of the wordlines 104 of the DRAM 100. The selection of one of the wordlines 104 causes the data stored in all of the memory cells 102 coupled to that wordline 104 to be loaded into the sense amplifiers (sense amps) 108. That data, or a portion thereof, may thereafter be placed onto the data bus 110 (for a read operation).

What portion of the data in the sense amps 108 is actually placed onto the data bus 110 in a read operation is determined by the column decoder 114. More specifically, the column decoder 114 receives a column address on the address lines 116, and a column address strobe (CAS) signal on a control line 120, and in response to these signals, the column decoder 114 decodes the column address to select one or more of the bitlines 106 of the DRAM 100. The number of bitlines 106 selected in response to a single column address may differ from implementation to implementation, and is referred to as the base granularity of the DRAM 100. For example, if each column address causes sixty-four bitlines 106 to be selected, then the base granularity of the DRAM 100 is eight bytes. Defined in this manner, the base granularity of the DRAM 100 refers to the amount of data that is read out of or written into the DRAM in response to each column address/CAS signal combination (i.e. each CAS or column command).

Once the appropriate bitlines 106 are selected, the data in the sense amps 108 associated with the selected bitlines 106 are loaded onto the data bus 110. Data is thus read out of the DRAM 100. Data may be written into the DRAM 110 in a similar

fashion. A point to note here is that in a typical DRAM, the address lines 116 are multiplexed. Thus, the same lines 116 are used to carry both the row and column addresses to the row and column decoders 112, 114, respectively. That being the case, a typical memory access requires at least two steps: (1) sending a row address on the address lines 116, and a RAS on the control line 118; and (2) sending a column address on the address lines 116, and a CAS on the control line 120.

A timing diagram illustrating the various steps carried out during typical DRAM read cycles is shown in Fig. 2. As shown, to initiate a read cycle, a row address 208(1) is placed onto the address lines 116, and a RAS signal 202(1) is asserted on the RAS control line 118. Then, a column address 210(1) is sent onto the address lines 116, and a CAS signal 204(1) is asserted on the CAS control line 118. A short time thereafter, the data 206(1) stored at the locations indicated by the row address 208(1) and the column address 210(1) appear on the data bus 110. Data is thus extracted from the DRAM 100. After the first set of data 206(1) disappears from the data bus 110, a second read operation may be initiated. Like the first read operation, the second read operation begins with a row address 208(2) on the address lines 116, and a RAS signal 202(2) on the RAS control line 118. Then, a column address 210(2) is sent onto the address lines 116, and a CAS signal 204(2) is asserted on the CAS control line 118. A short time thereafter, the data 206(2) stored at the locations indicated by the row address 208(2) and the column address 210(2) appear on the data bus 110. The second read operation is thus completed. Additional successive reads may be carried out in a similar fashion.

Notice from the timing diagram of Fig. 2 that, for individual read cycles, there is substantial idle time between successive data sets 206 on the data bus 110. During this

idle time, the data bus 110 is not utilized and no data is being transferred. The more idle time there is, the lower the utilization rate of the data bus 110, and the lower the utilization rate, the longer it will take for an external component (such as a CPU) to extract data from the DRAM 100. Since almost all operations of a computer require the use of memory, the longer it takes to get data from a memory, the slower the performance of the overall computer system. Thus, low bus utilization can have a direct negative impact on the overall performance of a computer system.

To improve data bus utilization, several techniques have been developed. One such technique involves the use of a "burst" mode of operation. Basically, in burst mode, rather than implementing just one column access for each RAS command, a plurality of column accesses are carried out for each RAS command. This results in consecutively accessing multiple sets of data from the same row of a DRAM 100. A timing diagram illustrating the operation of burst mode is shown in Fig. 3. More specifically, Fig. 3 depicts two burst mode read cycles, with each read cycle being directed to a different row of the DRAM.

To initiate a burst mode read cycle, a row address 308(1) is placed onto the address lines 116, and a RAS signal 302(1) is asserted on the RAS control line 118. Then, a column address 310(1) is sent onto the address lines 116, and a CAS signal 304(1) is asserted on the CAS control line 118. In response to the column address 310(1) and the CAS signal 304(1), the DRAM internally generates a plurality of additional column addresses. These additional column addresses are generated based upon the column address 310(1) that is provided, and a predetermined scheme. For example, the additional column addresses may be generated by incrementing the provided column

address 310(1), decrementing the column address 310(1), or by manipulating the column address 310(1) in some other manner. The number of additional column addresses generated by the DRAM depends upon the burst length that the DRAM is implementing. In the example shown in Fig. 3, the burst length is four; thus, three additional column addresses are generated by the DRAM.

As the provided column address 310(1) is received, and as each additional column address is generated, they are applied by the DRAM to access a particular set of data. These addresses are applied in succession so that multiple sets of data are accessed from the same row of the DRAM. A short time after the application of each column address, data 306 stored at the locations indicated by the row address 308(1) and the applied column address starts to appear on the data bus 110. Because this data 306 is extracted from the DRAM 100 in response to consecutive applications of column addresses, there is no idle time between the sets of data 306(1)-306(4) on the data bus 110. As a result, data bus utilization is improved.

After the first set of data 306(1)-306(4) disappears from the data bus 110, a second burst mode read operation may be initiated. Like the first read operation, the second read operation begins with a row address 308(2) on the address lines 116, and a RAS signal 302(2) on the RAS control line 118. Then, a column address 310(2) is sent onto the address lines 116, and a CAS signal 304(2) is asserted on the CAS control line 118. In response, the DRAM generates three additional column addresses, and applies the provided column address 310(2) and the additional column addresses in succession to access multiples set of data from the same row. Shortly after each column address is applied, data 306 stored at the locations indicated by the row address 308(2) and the

applied column address appears on the data bus 110. Because this data 306 is extracted from the DRAM 100 in response to consecutive applications of column addresses, there is again no idle time between the sets of data 306(5)-306(8) on the data bus 110. The second read operation is thus completed. Additional successive reads may be carried out

5 in a similar fashion.

Several aspects of burst mode operation should be noted. First, notice that burst mode significantly increases output data granularity. More specifically, because a burst mode memory request involves multiple column accesses, the data extracted from the DRAM in response to a burst mode request is not just one base granularity in size, but

10 rather is a multiple of the base granularity, where the multiple is equal to the burst length. Thus, in the example shown in Fig. 3, the output data granularity of the DRAM is four times the base granularity. This may pose a problem in some implementations. For example, in some applications, a CPU may wish to access only one base granularity of data at a time. If burst mode is implemented in such an application, then all of the data

15 after the first granularity will be dropped by the CPU. In such a case, even though data bus utilization is improved by the use of burst mode, overall system efficiency is not improved because the extra data from the memory is not used. From an efficiency point of view, the end result is the same as if burst mode were not implemented at all. In such applications, burst mode does not provide a useful solution.

20 A second aspect to note is that burst mode eliminates data bus idle time only so long as the same row is being accessed. As soon as a different row is accessed, a significant amount of idle time is introduced on the data bus 110, as shown in Fig. 3.

Thus, in applications where access of the DRAM switches from row to row on a regular

basis (as is often the case), there is a substantial amount of idle time on the data bus 110, even if burst mode is implemented.

To further improve data bus utilization, burst mode may be implemented in conjunction with a multi-bank DRAM to achieve full data bus utilization. In a multi-bank DRAM, the DRAM is divided into multiple "banks", which may be viewed as "virtual memories" within the DRAM. Each bank may be accessed individually, and each bank has its own set of sense amps. However, all banks share the same data bus. A block diagram of a sample multi-bank DRAM 400 is shown in Fig. 4. While Fig. 4 shows a DRAM 400 having four banks 404, it should be noted that more or fewer banks may be implemented if so desired. The basic concept behind a multi-bank DRAM 400 is that higher bus utilization may be achieved by "interleaving" or alternating memory requests between the different banks 404. By interleaving the memory requests, it is possible to initiate memory access to one bank (e.g. 404(1)) while another bank (e.g. 404(2)) is busy delivering data onto the data bus. By doing so, the data bus idle time of one bank is used advantageously by the other bank to put data onto the data bus 410. Because all banks 404 are using the same data bus 410, interleaving the memory requests in this way makes it possible to keep the data bus 410 constantly filled, even when different rows are being accessed.

To illustrate how burst mode and interleaving may be used to achieve full data bus utilization, reference will now be made to the timing diagram of Fig. 5. In Fig. 5, a RAS1 signal is used to indicate a RAS signal applied to bank 1 404(1), while a RAS2 signal is used to indicate a RAS signal applied to bank 2 404(2), and so on. Likewise, a CAS1

signal indicates a CAS signal being applied to bank 1 404(1), while a CAS2 signal indicates a CAS signal being applied to bank 2 404(2), and so on.

As shown in Fig. 5, a read operation from bank 1 404(1) of the DRAM 400 is initiated by first sending an asserted RAS signal 502(1) and a row address 508(1) to bank 1 404(1). Then, at a later time, an asserted CAS signal 504(1) and a column address 510(1) are sent to bank 1 404(1). A short time thereafter, data associated with the row address 508(1) and the column address 510(1) are sent onto the data bus 410 (Fig. 4) by the sense amps 408(1) of bank 1 404(1). In the timing diagram shown in Fig. 5, it is assumed that bank 1 404(1) implements a burst mode length of two. Thus, in response to the one column address 510(1), two sets of data 506(1), 506(2) are outputted onto the data bus 410 by bank 1 404(1).

After the RAS signal 502(1) is sent to bank 1 404(1) but before the CAS signal 504(1) is sent to bank 1 404(1), an asserted RAS signal 502(2) and a row address 508(2) are sent to bank 2 404(2) of the DRAM 400. In addition, an asserted CAS signal 504(2) and a column address 510(2) are sent to bank 2 404(2) at a later time. In response to these signals, bank 2 404(2) outputs data associated with the row address 508(2) and the column address 510(2) onto the data bus 410 using the sense amps 408(2). As was the case with bank 1 404(1), bank 2 404(2) also implements a burst mode length of two. As a result, two sets of data 506(3), 506(4) are outputted onto the data bus 410 by bank 2 404(2) in response to the one column address 510(2). These sets of data 506(3), 506(4) immediately follow the sets of data 506(1), 506(2) outputted by bank 1; thus, there is no idle time between the data sets.

After the RAS signal 502(2) is sent to bank 2 404(2) but before the CAS signal 504(2) is sent to bank 2 404(2), an asserted RAS signal 502(3) and a row address 508(3) are sent to bank 3 404(3). A short time thereafter, an asserted CAS signal 504(3) and a column address 510(3) are sent to bank 3 404(3). In response, bank 3 404(3) outputs data associated with row address 508(3) and column address 510(3) onto the data bus 410 using the sense amps 408(3). As was the case with bank 1 and bank 2 404(2), bank 3 404(3) also implements a burst mode length of two. As a result, two sets of data 506(5), 506(6) are outputted onto the data bus 410 by bank 3 404(3) in response to the one column address 510(3). These sets of data 506(5), 506(6) immediately follow the sets of data 506(3), 506(4) outputted by bank 2; thus, there is no idle time between the data sets.

To finish the example, after the RAS signal 502(3) is sent to bank 3 404(3) but before the CAS signal 504(3) is sent to bank 3 404(3), an asserted RAS signal 502(4) and a row address 508(4) are sent to bank 4 404(4). A short time later, an asserted CAS signal 504(4) and a column address 510(4) are sent to bank 4 404(4). In response, bank 4 404(4) outputs data associated with row address 508(4) and column address 510(4) onto the data bus 410 using the sense amps 408(4). As was the case with the other banks, bank 4 404(4) implements a burst mode length of two. As a result, two sets of data 506(7), 506(8) are outputted onto the data bus 410 by bank 4 404(4) in response to the one column address 510(4). These sets of data 506(7), 506(8) immediately follow the sets of data 506(5), 506(6) outputted by bank 3; thus, there is no idle time between the data sets.

While bank 4 404(4) is being accessed, access of bank 1 404(1) may again be initiated with a RAS signal and a row address, as shown, to extract more data from that bank. This process of interleaving accesses between the various banks 404 may continue

indefinitely to continually access data from the DRAM 400. By combining burst mode operation with a multi-bank DRAM 400 as shown in this example, it is possible to achieve full data bus utilization.

In practice, a certain number of banks are needed to achieve full data bus utilization in a particular DRAM, where the number of banks needed is determined by certain timing parameters of the DRAM. One relevant timing parameter is the minimum time required between consecutive RAS signals to the same bank. This parameter, denoted herein as T_{rc} , is often referred to as the RAS cycle time. Another relevant parameter is the amount of time it takes to place one base granularity of data onto the data bus. This parameter is denoted herein as Dt . To determine the number of banks needed to achieve full data bus utilization, T_{rc} is divided by $n \cdot Dt$ where n is the burst length. If, for example, T_{rc} is 80ns and Dt is 10ns and the DRAM is implementing a burst length of two, then the number of banks needed is 80ns/20ns or four. With these timing parameters (which are typical) and four banks, a DRAM can achieve full data bus utilization.

While the combination of burst mode and a multi-bank DRAM 400 makes it possible to achieve 100% data bus utilization in a memory, this implementation does not come without its drawbacks. One significant drawback is that it still relies upon burst mode to achieve full data bus utilization. Because of this reliance, this implementation suffers from the same shortcoming as that experienced in regular burst mode. Namely, it increases the data granularity of the DRAM 400. Notice from the timing diagram of Fig. 5 that instead of outputting just one base granularity of data per access to each bank, the DRAM 400 outputs two (it is two in the example shown in Fig. 5 but it could more than two in other implementations). This increase in data granularity can lead to inefficiency.

As noted previously, in some applications, an external component (such as a CPU) may wish to access only one base granularity of data at a time. In such applications, any data provided after the first base granularity will be dropped. If burst mode is implemented in such an application, at least half of the data provided by the DRAM 400 will be dropped, which means that at most 50% efficiency can be achieved. Thus, even though burst mode combined with a multi-bank DRAM 400 may achieve 100% data bus utilization in such an application, it does not improve overall system efficiency because the extra data is not used. Consequently, the burst mode/multi-bank DRAM combination does not provide a complete solution for all possible applications.

As an alternative to the burst mode/multi-bank DRAM combination, a plurality of separate DRAM's may be implemented to achieve full data bus utilization. By interleaving memory requests between separate DRAM's instead of between separate banks within a single DRAM, it is possible to achieve full data bus utilization without requiring an increase in data granularity. This result comes with extra cost and complexity, however. To illustrate how separate DRAM's may be used to achieve full data bus utilization, reference will be made to Figs. 6 and 7. Specifically, Fig. 6 shows a block diagram of a sample multi-DRAM implementation, while Fig. 7 shows a timing diagram for several read cycles of the implementation of Fig. 6.

As shown in Fig. 6, the sample implementation comprises a plurality of separate DRAM's 604(1), 604(2) (more than two may be implemented if so desired), with each DRAM 604 having its own separate command lines 608(1), 608(2), and address lines 606(1), 606(2). Both DRAM's share the same data bus 620, and both have 2 banks. In addition to the DRAM's 604(1), 604(2), the implementation further comprises a controller

602 for controlling the interleaving of requests between the DRAM's 604(1), 604(2). It is the responsibility of this controller 602 to manage the interleaving of memory requests such that: (1) the data bus 620 is used as fully as possible; and (2) there is no bus contention on the data bus 620.

5 To illustrate how the implementation of Fig. 6 can be used to achieve full data bus utilization, reference will be made to the timing diagram of Fig. 7. For clarity purposes, RAS1,1 is used in Fig. 7 to indicate a RAS command applied to DRAM 1 604(1), bank 1, while RAS2,1 is used to indicate a RAS command applied to DRAM 2 604(2), bank 1, and so on. Likewise, CAS1,1 is used to indicate a CAS command being applied to
10 DRAM 1 604(1), bank 1, while CAS2,1 is used to indicate a CAS command being applied to DRAM 2 604(2), bank 1.

As shown in Fig. 7, to extract data from the DRAM's 604(1), 604(2), the controller 602 first initiates a read operation on DRAM 1 604(1), bank 1. This is carried out by sending a RAS command 702(1) and a row address 706(1) to DRAM 1 604(1),
15 bank 1. Then, a CAS command 704(1) and a column address 706(2) are sent to DRAM 1 604(1), bank 1. A short time thereafter, data 720(1) associated with the row address 706(1) and the column address 706(2) are outputted onto the data bus 620 (Fig. 6) by DRAM 1 604(1), bank 1. While the CAS command 704(1) and the column address 706(2) are being sent to DRAM 1 604(1), bank 1, the controller 602 also sends a RAS
20 command 712(1) and a row address 716(1) to DRAM 2 604(2), bank 1. Thereafter, a CAS command 714(1) and a column address 716(2) are sent to DRAM 2 604(2), bank 1. In response to these signals, DRAM 2 604(2), bank 1, outputs data 720(2) associated with the row address 716(1) and the column address 716(2) onto the data bus 620. This set of

data 720(2) immediately follows the set of data 720(1) outputted by DRAM 1 604(1), bank 1; thus, there is no idle time between the data sets.

While the CAS signal 714(1) and the column address 716(2) are being sent to DRAM 2 604(2), bank 1, a RAS command 702(2) and row address 706(3) are sent to
 5 DRAM 1 604(1), bank 2, to initiate another read cycle. Thereafter, a CAS command 704(2) and a column address 706(4) are sent to DRAM 1 604(1), bank 2. In response, DRAM 1 604(1), bank 2, outputs data 720(3) associated with the row address 706(3) and the column address 706(4) onto the data bus 620. This set of data 720(3) immediately follows the set of data 720(2) outputted by DRAM 2 604(2), bank 1; thus, there is again
 10 no idle time between the data sets. To finish the example, while the CAS command 704(2) and the column address 706(4) are being sent to DRAM 1 604(1), bank 2, the controller 602 initiates a read cycle on DRAM 2 604(2), bank 2, by sending a RAS command 712(2) and a row address 716(2) to DRAM 2 604(2), bank 2. Thereafter, a CAS command 714(2) and a column address 716(4) are sent to DRAM 2 604(2), bank 2.
 15 In response to these signals, DRAM 2 604(2), bank 2, outputs data 720(4) associated with the row address 716(3) and the column address 716(4) onto the data bus 620. This set of data 720(4) immediately follows the set of data 720(3) outputted by DRAM 1 604(1), bank 2; hence, there is no idle time between the data sets. Additional read operations may be carried out in this manner to continue extracting data from the DRAM's 604(1),
 20 604(2). As this example illustrates, by interleaving memory requests between multiple banks of multiple DRAM's, it is possible to achieve full data bus utilization, and because no burst mode is implemented, data granularity is not increased.

While the multi-DRAM implementation is able to achieve full data bus utilization without increasing data granularity, it does so at a significant cost. First, due to the tight timing constraints, the DRAM's shown in Fig. 6 are very fast and very expensive. With multiple DRAM's being required to implement the system of Fig. 6, the cost of the memory system can be prohibitive. Compared to single DRAM memory systems, the cost of this multi-DRAM system can be several-fold. Also, the multi-DRAM implementation is limited in its application. By its very nature, it can be implemented only in a multi-DRAM environment. In the many applications in which it is desirable to implement just one DRAM, the multi-DRAM implementation cannot be used. In addition, this implementation can add substantial complexity to the memory access process. Because the controller 602 must concurrently control multiple DRAM's, the memory access process is much more complex and difficult to manage than in a single DRAM implementation. Overall, there is a significant price to pay for the functionality provided by the multi-DRAM configuration, and in many implementations, this price is prohibitive. Hence, the multiple DRAM approach does not provide a viable solution for all applications.

Summary

In view of the shortcomings of the prior art, the present invention provides an improved mechanism, which enables full data bus utilization to be achieved within a memory, but which does not increase the data granularity of the memory or require multiple memories to be implemented. In accordance with one embodiment, there is provided a memory comprising a first memory portion, a second memory portion, and an

interface. To enable the memory portions to be manipulated independently, the memory portions in one embodiment are electrically isolated from each other. In one embodiment, this electrical isolation is achieved by physically placing the interface between the memory portions. In addition, the interface is adapted, in one embodiment, to enable it to receive a row command and a column command in the time it takes to cycle the memory once (to read one base granularity of data out of or write one base granularity of data into the memory). With such a memory, it is possible to achieve full data bus utilization without increasing data granularity.

In one embodiment, full data bus utilization is achieved by interleaving access requests between the two memory portions. More specifically, the interface receives a first access request (comprising a row command and no more than one column command) on a set of control ports, and forwards the request on to the first memory portion for processing. In response, the first memory portion accesses a first set of data having a size of no more than one base granularity, and sends that data onto a data bus (assuming a read operation). A data set is thus extracted from the memory. Thereafter, the interface receives a second access request on the same set of control ports, and forwards the request to the second memory portion for processing. In response, the second memory portion accesses a second set of data also having a size of no more than one base granularity, and sends that data onto the data bus. If the access requests are timed properly relative to each other, then the second set of data will be sent onto the data bus immediately after the first set of data such that there is substantially no idle time on the data bus. With no data bus idle time, full data bus utilization is achieved. This full data bus utilization may be perpetually maintained by continuously interleaving access

requests between the two portions. In one embodiment, proper relative timing between the various access requests is made possible, as least partially, by the fact that the memory portions are electrically isolated from each other, and hence, may be independently manipulated, and by the fact that a row command and a column command can be received
 5 by the interface in the time it takes to cycle either memory portion once.

Notice that full data bus utilization is achieved without increasing data granularity. More specifically, notice that burst mode does not need to be implemented, and that no more than one base granularity of data is sent onto the data bus in response to each access request. In addition, notice that full data bus utilization is achieved without
 10 implementing multiple memories. The different memory portions may behave like independent memories, but they are both part of the same memory using the same set of control ports. Thus, multiple memories need not be implemented. By achieving full data bus utilization without increasing data granularity and without implementing multiple memories, the present invention provides a significant advance over the prior art.

Brief Description of the Drawings

Fig. 1 is a functional diagram of a typical DRAM.

Fig. 2 is a timing diagram illustrating the steps carried out by the DRAM of Fig. 1 during several read cycles.

20 Fig. 3 is a timing diagram illustrating the steps carried out during several burst mode read cycles.

Fig. 4 is a block diagram of a sample multi-bank DRAM.

Fig. 5 is a timing diagram illustrating the operation of the DRAM of Fig. 4 during several read cycles.

Fig. 6 is a block diagram of a multiple DRAM implementation.

Fig. 7 is a timing diagram illustrating the operation of the implementation shown
5 in Fig. 6 during several read cycles.

Fig. 8 is a block diagram of a system in which one embodiment of the present invention may be implemented.

Fig. 9 is a detailed block diagram of a memory in accordance with one embodiment of the present invention.

Fig. 10 is a timing diagram illustrating the operation of the memory shown in Fig.
10 9 during several read cycles.

Fig. 11 is a timing diagram illustrating an alternative operation of the memory of Fig. 9 wherein a single set of control ports/lines is used to send both row and column commands.

Fig. 12 is a flow diagram illustrating the operation of the memory controller of
15 Fig. 8 in accordance with one embodiment of the present invention.

Detailed Description of Embodiment(s)

With reference to Fig. 8, there is shown a block diagram of a system 800 in which
20 one embodiment of the present invention may be implemented, the system 800 comprising a memory 802, a memory controller 804, and an external component 806. For purposes of illustration, it will be assumed in the following discussion that memory 802 is a dynamic random access memory (DRAM). However, it should be noted that the teachings of the

present invention may be applied to other types of memory as well, if so desired. In one embodiment, memory 802 is implemented as an integrated circuit.

In system 800, the external component 806 (which may, for example, be a CPU executing a program) is the component that requires access to the memory 802 for purposes of writing data into the memory 802, reading data out of the memory 802, or both. To access the memory 802, the external component 806 submits requests to the memory controller 804 via a set of control 830 and data lines 832. In response, the memory controller 804 translates the requests into access requests that the memory 802 can understand, and sends the access requests to the memory 802 via another set of control 822 and data lines 820. In the case of a write, data is provided to the memory 802 via the data lines 820. In the case of a read, data is provided by the memory 802 to the controller 804 via the data lines 820.

The interface 810 of the memory 802 receives the access requests from the controller 804 and responds by accessing the memory portions 812(1), 812(2) of the memory 802. In one embodiment, the interface 810 accesses whichever portion 812(1), 812(2) is indicated in an access request. Since it is the controller 804 that generates the access requests, it is the controller 804 that controls which portions 812(1), 812(2) of the memory 802 are accessed and in what sequence. As will be discussed in a later section, it is desirable to interleave access requests between the different portions 812(1), 812(2) to achieve full data bus utilization. Since it is the controller 804 that controls which memory portions 812(1), 812(2) are accessed, it is up to the controller 804 to properly interleave the access requests between the portions 812(1), 812(2) to enable full data bus utilization. The operation of the controller 804 will be described in greater detail in a later section. In the

embodiment shown in Fig. 8, the memory 802 is depicted as comprising two portions 812(1), 812(2). It should be noted that this is for illustrative purposes only. If so desired, the memory 802 may be implemented with any number of portions. Each portion 812 may comprise one or more banks. In the following description, for the sake of simplicity, it will be assumed that each portion 812 comprises one bank. However, it should be noted that for purposes of the present invention, each portion 812 may comprise 1 to n banks, where n is any integer.

With reference to the block diagram of Fig. 9, an embodiment of the memory 802 will now be described in greater detail. As shown in Fig. 9, the memory 802 comprises an interface 810, and two memory portions 812(1), 812(2). The first memory portion 812(1) comprises a bank with a plurality of memory/storage cells 902(1) arranged in rows and columns, and a row decoder 904(1) and column decoder 906(1) for facilitating access to the memory cells 902(1). In addition, the first memory portion 812(1) comprises a set of sense amps 908(1) coupled to a data bus 820 for receiving data therefrom and sending data thereto. Similarly, the second memory portion 812(2) comprises a bank with a plurality of memory/storage cells 902(2) arranged in rows and columns, and a row decoder 904(2) and column decoder 906(2) for facilitating access to the memory cells 902(2). In addition, the second memory portion 812(2) comprises a set of sense amps 908(2) coupled to the data bus 820 for receiving data therefrom and sending data thereto.

In one embodiment, the memory portions 812(1), 812(2) may be treated like independent memories. That is, each portion 812 may be accessed independently of the other portion 812 from a timing standpoint. More specifically, portion 812(1) may be accessed at any time relative to the accessing of portion 812(2), and portion 812(2) may be

accessed at any time relative to the accessing of portion 812(1). Put another way, no timing constraint is imposed on when one portion may be accessed relative to the other. Thus, there is no minimum required time delay between accessing one portion 812(1) and accessing the other portion 812(2). This is quite different from typical DRAM's, which usually require a time delay between successive accesses to the same DRAM, even if it is to different banks. As will be explained further below, this "independent" nature of the memory portions 812 greatly facilitates the process of interleaving access requests between the portions 812(1), 812(2) to achieve full data bus utilization without increasing data granularity. At this point, it should be noted that while the memory portions 812(1), 812(2) may be treated like independent memories from an accessing point of view, they are still part of the same memory 802. They share the same data bus 820, the same interface 810, and the same control ports/lines 822. Thus, unlike the multi-DRAM implementation discussed previously, the memory 802 is a single memory, not multiple, separate memories.

In one embodiment, the independent nature of the memory portions 812 is derived from that fact that they are electrically isolated from each other. By electrically isolated, it is meant that the two portions 812(1), 812(2) are sufficiently decoupled from each other from an electrical noise standpoint that the access of one portion does not corrupt data in the other portion. More specifically, the activation of one set of sense amps 908 does not corrupt data in the other set of sense amps 908, regardless of the timing of the activations.

As a result of this electrical isolation, it is possible to activate the sets of sense amps 908(1), 908(2) independently to process access requests without fear that processing an access request in one portion 812 will adversely affect the processing of an access request in the other portion.

In one embodiment, the electrical isolation between the memory portions 812(1), 812(2) is achieved by physically placing the interface 810 between the two portions 812(1), 812(2). The interface 810 provides a natural and convenient barrier between the two portions 812(1), 812(2) and works effectively to electrically isolate one portion 812(1) from the other 812(2). It should be noted that this is just one possible way of achieving electrical isolation between the portions 812(1), 812(2). The same result may be achieved in many other ways (e.g. improved sense amp circuits, better power supply control, circuit enhancements, etc.). All such means for electrically isolating the portions 812(1), 812(2) are within the scope of the present invention.

As noted above, both memory portions 812(1), 812(2) share the interface 810 and the control ports/lines 822. In one embodiment, the interface 810 receives all access requests via the control ports/lines 822 and forwards the requests on to the proper memory portion 812(1), 812(2) for processing. In effect, the interface 810 acts as a manager to manage the processing of access requests by the memory portions 812(1), 812(2). In carrying out its management function, the interface 810 receives several sets of information on the control ports/lines 822 from the controller 804 for each access request. In one embodiment, these sets of information include: (1) portion selection information 924 that specifies which memory portion 812(1), 812(2) is to be accessed; (2) row control information 920 that specifies which row of storage cells within a memory portion is to be accessed; and (3) column control information 922 that specifies which column or columns of storage cells within that row are to be accessed. Together, these sets of information provide all of the information needed to access one base granularity of data from one of the portions 812(1), 812(2) of the memory 802.

With regard to the row and column control information 920, 922, these sets of information may be sent in any desired format. For example, the row control information 920 may take the form of a row command (e.g. comprising a row address and a RAS) sent as a set of parallel bits, or it may take the form of a row command sent as a serialized packet, or it may take any other form. So long as sufficient information is provided to the interface 810 to instruct the interface 810 to access a particular row of storage cells, any row command format may be used. The same is true for the column control information 922. Specifically, the column control information 922 may take the form of a column command (e.g. comprising a column address and a CAS) sent as a set of parallel bits, or it may take the form of a column command sent as a serialized packet, or it may take any other form. So long as sufficient information is provided to the interface 810 to instruct the interface 810 to access one or more particular columns of storage cells, any column command format may be used.

In response to the control information on the control ports/lines 822, the interface 810 manages the operation of the memory portions 812(1), 812(2). In one embodiment, the interface 810 is capable of receiving both a row command and a column command from the controller 804 in an amount of time X, where X is less than or equal to the amount of time T it takes to cycle either of the memory portions 812(1), 812(2) once. Used in this context, the amount of time T it takes to cycle the memory 802 is the amount of time needed to read one base granularity of data out of, or write one base granularity of data into either portion 812(1), 812(2) of the memory 802. As defined previously, the base granularity of the memory 802 refers to the amount of data that is read out of or written into the memory 802 in response to each column command (when no burst mode is implemented). With this

ability to receive both a row command and a column command in the time it takes to cycle the memory 802 once, the interface 810 greatly facilitates the process of achieving full data bus utilization without increasing data granularity. As will be explained more fully in a later section, the row command and the column command received by the interface 810 in the amount of time X may correspond to different access requests due to the process of pipelining.

To illustrate how the memory 802 may be used advantageously to achieve full data bus utilization, reference will now be made to the timing diagram of Fig. 10, which shows eight read access requests interleaved between the two portions 812(1), 812(2). For the sake of simplicity, Fig. 10 shows only read cycles. However, it should be noted that write operations may be carried out in a similar fashion. For purposes of explanation, the following nomenclature will be used in Fig. 10: (1) P1 will be used to indicate an access request to the first portion 812(1) of the memory, while P2 will be used to indicate an access request to the second portion 812(2); (2) Row1 will be used to indicate a row command to portion 812(1), while Row2 will be used to indicate a row command to portion 812(2); and (3) Col1 will be used to indicate a column command to portion 812(1), while Col2 will be used to indicate a column command to portion 812(2). With this in mind, the operation of the memory 802 will now be described.

Initially, the interface 810 receives a first access request from the controller 804 to read a set of data from the first portion 812(1) of the memory 802. This first access request comprises a portion indication 1002(1) sent on the portion control port/line 924, a row command 1004(1) sent on the row control ports/lines 920, and a column command 1006(1) sent on the column control ports/lines 922. In the embodiment shown, the row command

1004(1) is sent first and the column command 1006(1) is sent later. In a synchronous DRAM (SDRAM), the row command 1004(1) and the column command 1006(1) may be sent in consecutive clock cycles. Upon receiving this access request, the interface 810 determines that it is intended for the first memory portion 812(1); hence, it forwards the
 5 row command 1004(1) and the subsequent column command 1006(1) on to portion 812(1).

In response, the row decoder 904(1) of the first memory portion 812(1) decodes the row address contained in the row command 1004(1). This causes one of the rows of storage cells 902(1) to be accessed, and the data contained therein to be loaded into the sense amps 908(1). In addition, the column decoder 906(1) of the first portion 812(1)
 10 decodes the column address contained in the column command 1006(1), which causes a subset of the data contained in the sense amps 908(1) to be sent onto the data bus 820. This output data 1008(1) appears on the data bus 820 a short time after the first access request is received, and is one base granularity in size (because the access request contained only one column command and burst mode was not implemented). A first set of data 1008(1) is thus
 15 extracted from the memory 802.

While the first access request is being processed, the interface 810 receives a second access request from the controller 804 to read a set of data from the second portion 812(2) of the memory 802. This second access request comprises a portion indication 1002(2) sent on the portion control port/line 924, a row command 1004(2) sent on the row control
 20 ports/lines 920, and a column command 1006(2) sent on the column control ports/lines 922. As was the case with the first access request, the row command 1004(2) is sent first and the column command 1006(2) is sent later. Notice from the embodiment shown in Fig. 10 that the row command 1004(2) of the second access request is sent concurrently with the

column command 1006(1) of the first access request. This pipelining of the different parts of different access requests facilitates the process of interleaving access requests to the different memory portions 812. In one embodiment, both the row command 1004(2) and the column command 1006(1) are received in an amount of time X, which is less than or equal to the time T needed to cycle either of the memory portions 812(1), 812(2) once. The ability of the controller 804 to send, and the ability of the interface 810 to receive, both a row command and a column command in an amount of time X less than or equal to T contributes to the system's ability to achieve full data bus utilization without increasing data granularity.

Another point to note in the example shown in Fig. 10 is that the row command 1004(2) of the second access request is received immediately after the row command 1004(1) of the first access request. There is no required minimum time delay between the two commands 1004(1), 1004(2). In an SDRAM, the row commands 1004(1), 1004(2) may be received in consecutive clock cycles. In one embodiment, this is made possible by the fact that the two memory portions 812(1), 812(2) are electrically isolated from each other, and hence, may be treated like independent memories. Because the two memory portions 812(1), 812(2) may be treated like independent memories, it is possible to receive and forward an access request to one of the memory portions 812(1), 812(1) without regard to the other. This in turn means that there is no timing restriction on when a row command may be sent to one memory portion relative to a row command being sent to the other memory portion. As will be discussed further below, this lack of a timing constraint between the row commands facilitates the process of achieving full data bus utilization without increasing data granularity.

Upon receiving the second access request, the interface 810 determines that it is intended for the second memory portion 812(2); hence, it forwards the request on to portion 812(2). In response, the row decoder 904(2) of the second memory portion 812(2) decodes the row address contained in the row command 1004(2). This causes one of the rows of storage cells 902(2) to be accessed, and the data contained therein to be loaded into the sense amps 908(2). In addition, the column decoder 906(2) of the second portion 812(2) decodes the column address contained in the column command 1006(2), which causes a subset of the data contained in the sense amps 908(2) to be sent onto the data bus 820. This output data 1008(2) appears on the data bus 820 a short time after the second access request is received, and like the first set of data 1008(1), is one base granularity in size. If the second access request is properly timed relative to the first access request, then the second set of data 1008(2) will be sent onto the data bus 820 immediately after the first set of data 1008(1) so that there is substantially no idle time between the data sets 1008(1), 1008(2). With no data bus idle time between the data sets 1008(1), 1008(2), full data bus utilization is achieved.

While the second access request is being processed, the interface 810 receives a third access request, this one being directed back to the first memory portion 812(1). Like the first two access requests, the third access request comprises a portion indication 1002(3), a row command 1004(3), and a column command 1006(3). The row command 1004(3) is received first and the column command 1006(3) is received later. As shown, the row command 1004(3) is received immediately after the row command 1004(2) of the second access request and is received concurrently with the column command 1006(2) of the second access request. The row command 1006(3) and the column command 1006(2)

are received in an amount of time X, which is less than or equal to the amount of time T needed to cycle either of the memory portions 812(1), 812(2) once.

Upon receiving this third access request, the interface 810 determines that it is intended for the first memory portion 812(1); hence, it forwards the request on to portion 812(1). In response, the row decoder 904(1) and the column decoder 906(1) of the first
 5 portion 812(1) decode the row address and the column address contained in the row command 1006(3) and column command 1006(3), respectively, and cause a base granularity of data to be sent from the sense amps 908(1) onto the data bus 820 (in the manner already described). This output data 1008(3) appears on the data bus 820 a short
 10 time after the third access request is received, and like the first two sets of data 1008(1), 1008(2), is one base granularity in size. If the third access request is properly timed relative to the second access request, then the third set of data 1008(3) will be sent onto the data bus 820 immediately after the second set of data 1008(2). Thus, there is substantially no idle time between the data sets 1008(2), 1008(3), which in turn enables full data bus utilization
 15 to be achieved.

While the third access request is being processed, the interface 810 receives a fourth access request, this one being directed back to the second memory portion 812(2). Like the previous requests, the fourth access request comprises a portion indication 1002(4), a row command 1004(4), and a column command 1006(4). The row command 1004(4) is
 20 received first and the column command 1006(4) is received later. As shown, the row command 1004(4) is received immediately after the row command 1004(3) of the third access request and is received concurrently with the column command 1006(3) of the third access request. The row command 1006(4) and the column command 1006(3) are received

in an amount of time X, which is less than or equal to the amount of time T needed to cycle either of the memory portions 812(1), 812(2) once.

Upon receiving this fourth access request, the interface 810 determines that it is intended for the second memory portion 812(2); hence, it forwards the request on to portion 812(2). In response, the row decoder 904(2) and the column decoder 906(2) of the second
 5 portion 812(2) decode the row address and the column address contained in the row command 1006(4) and column command 1006(4), respectively, and cause a base granularity of data to be sent from the sense amps 908(2) onto the data bus 820 (in the manner already described). This output data 1008(4) appears on the data bus 820 a short
 10 time after the third access request is received, and like the previous sets of data 1008(1), 1008(2), 1008(3), is one base granularity in size. If the fourth access request is properly timed relative to the third access request, then the fourth set of data 1008(4) will be sent onto the data bus 820 immediately after the third set of data 1008(3). Thus, there is substantially no idle time between the data sets 1008(3), 1008(4), which in turn enables full
 15 data bus utilization to be achieved. In the manner described, access requests may be continuously interleaved between the portions 812(1), 812(2) of the memory 802 to achieve sustained full data bus utilization, and since neither burst mode nor a multiple memory configuration is implemented, this full data bus utilization is achieved without increasing data granularity and without implementing multiple memories. Thus, full data bus
 20 utilization is achieved without the shortcomings of the prior art.

To facilitate a complete understanding of the invention, a comparison will now be made between the timing diagram of Fig. 10 and the timing diagram of Fig. 5, which shows the operation of the prior art multi-bank DRAM of Fig. 4. Several points should be noted

with regard to the timing diagram of Fig. 5. First, note that in Fig. 5, the DRAM 402 requires more time to receive a complete access request than it does to place one granularity of data 506 onto the data bus 410. More specifically, to receive a complete access request, the DRAM 402 has to: (1) receive a RAS signal and a row address; and then (2) receive a CAS signal and a column address. In that same amount of time, two base granularities of data can be accessed and placed onto the data bus 410 (i.e. the DRAM 402 can be cycled twice), as shown. Because the DRAM 402 takes longer to receive a complete request than it does to cycle the memory once, the DRAM 402 cannot keep the data bus 410 full unless it accesses several base granularities of data in response to each access request (i.e. implements burst mode). Unfortunately, by implementing burst mode, the DRAM 402 increases its output data granularity. As discussed previously, this has undesirable consequences.

The memory 802 of Fig. 9 has no such problem. Because the controller 804 is able to send, and the interface 810 of memory 802 is able to receive, both a row command and a column command (even though they may correspond to different access requests) in the time it takes to cycle the memory 802 once, memory 802 is not required to implement burst mode. Hence, memory 802 is able to achieve full data bus utilization without increasing data granularity, as shown in Fig. 10.

Another point to note with regard to Fig. 5 is that DRAM 402 has a time constraint, commonly referred to as T_{rr} , which precludes it from receiving one RAS signal (i.e. a row command) immediately after another. More particularly, the DRAM 402 is required to wait at least a T_{rr} period of time between successive RAS signals, even if the RAS signals are

directed to different banks. This time constraint is due, at least in part, to the fact that the banks 404(1), 404(2) of DRAM 402 are not electrically isolated from each other.

To elaborate, when a row of memory cells is accessed in response to a row command, one of the sets of sense amps 408 is activated. The activation of one of the sets of sense amps 408 causes a current spike, which in turn causes electrical noise. In a typical multi-bank DRAM 402, there is no electrical isolation between the various banks 404; hence, the noise generated in one set of sense amps 408 is felt in the other. To prevent the noise from one set of sense amps 408 from corrupting the data in the other set of sense amps 408, it is typically necessary to wait a T_{rr} period of time between successive sense amp activations. Because of this time constraint, there is a minimum required time delay of T_{rr} between successive row commands. This is so even if the row commands are directed to different banks 404 of the DRAM 402. As a result of this time constraint, the time period between successive row commands is lengthened. With this lengthened time period, it is very difficult if not impossible for the DRAM 402 to receive a complete access request in the time it takes to cycle the DRAM 402 once. As a result, DRAM 402 is unable to achieve full data bus utilization without implementing burst mode.

Again, the memory 902 of Fig. 9 has no such problem. Because the memory portions 812(1), 812(2) are electrically isolated from each other, noise from one portion 812 does not affect the other. As a result, there is no required minimum time delay between successive sense amp activations in the different portions 812, which in turn means that there is no required minimum time delay between successive interleaved row commands. As shown in Fig. 10, interleaved row commands 1004 may be received and passed on to the different memory portions 812 in immediate succession, if so desired. By eliminating the

Trr time constraint, memory 802 makes it possible to time the access requests such that data
 sets 1008 are placed onto the data bus 820 in immediate succession with substantially no
 intervening idle time. By eliminating data bus idle time, full data bus utilization is
 achieved. As noted previously, this is accomplished without increasing data granularity and
 5 without implementing multiple separate memories. Hence, memory 802 represents a
 significant improvement over the prior art.

Thus far, the operation of the system 800 has been described only with reference to
 read operations. It should be noted, though, that the same concepts, timing, sequence of
 commands, and data ordering may be applied to write operations. The main difference is
 10 that instead of the memory 802 sending data onto the data bus 820 to the controller 804, the
 controller 804 is sending data onto the data bus 820 to the memory 802. More specifically,
 in a read operation, the memory 802: (1) accesses a row of data from one of the memory
 portions 812; (2) loads that data into one of the sets of sense amps 908; and (3) outputs a
 portion of the data in the sense amps 908 onto the data bus 820. In a write operation, the
 15 memory 802: (1) accesses a row of data from one of the memory portions 812; (2) loads
 that data into one of the sets of sense amps 908; (3) obtains a set of write data from the data
 bus 820; (4) loads the write data into a portion of one of the sets of sense amps 908; and (4)
 stores the data in the sense amps 908 back into the appropriate row. As this discussion
 shows, the difference between a read and a write operation resides mainly in the direction
 20 of the flow of data. With regard to the timing of the portion control signals, the row and
 column commands, and even the data sets on the data bus, all may remain the same as that
 shown in Fig. 10 for read operations. In a write operation, it will be up to the controller 804
 to send the data onto the data bus 820 at the proper time.

Thus far, memory 802 has been described as having separate row control 920 and column control 922 ports/lines, and as receiving row and column commands concurrently on those ports/lines. While this is one possible embodiment, it should be noted that other embodiments are also possible. For example, if so desired, one set of control lines may be multiplexed to carry both the row command and the column command, as shown in the timing diagram of Fig. 11. This may be carried out, for example, by running the single set of control lines at double speed, or by sending a row command and a column command on different halves of a clock cycle. So long as a row command and a column command can be received in the time it takes to cycle the memory 802 once, any method/configuration for supplying the row command and column command may be used/implemented.

As note previously, in one embodiment, it is the memory controller 804 (Fig. 8) that controls the accessing of data from the memory 802. In particular, it is the controller 804 that controls the sending of access requests to the memory 802, and the interleaving of access requests to the different portions 812 of the memory 802 (as shown in Fig. 10). In carrying out this function, the controller 804 has the ability to send a row command and a column command to the memory 802 in an amount of time X, where X is less than or equal to the amount of time T that it takes to cycle the memory once, and has the logic for properly controlling the timing of the access requests.

In controlling the timing of the access requests, the controller 804 in one embodiment implements the following rule. If two consecutive access requests are directed to different portions 812 of the memory 802, then the access requests may be timed such that they cause the data bus 820 to be fully utilized (i.e. no data bus idle time between successive data sets, as shown in Fig. 10). Because the access requests are directed to

different portions 812 of the memory 802, there is no required minimum time delay between the access requests (i.e. between the row commands of the access requests).

On the other hand, if two consecutive requests are directed to the same memory portion 812, then the controller 804 imposes a delay between the row commands of the access requests, if necessary. This delay may, for example, be equal to T_{rr} . Because the access requests will cause the same set of sense amps to be activated, this delay is implemented to prevent data errors. The operation of the controller 804 is shown in the flow diagram of Fig. 12.

In operation, the controller 804 receives (1202) a request from the external component 806 to access the memory 802. In response, the controller 804 translates (1204) the request to derive an access request that the memory 802 can understand. In one embodiment, based upon the request, the controller 804 generates a set of portion control information, a row command, and a column command. Based upon the access request, the controller 804 determines (1206) the memory portion 812 that needs to be accessed to service this request. If this portion 812 is determined (1208) by the controller to be the same portion as that accessed by the access request immediately preceding the current access request, then the controller 804 imposes (1210) a time delay, if necessary. More specifically, the controller 804 determines whether a certain amount of time, such as T_{rr} , has passed since the row command of the previous access request was sent to the memory 802. If not, then a delay is imposed to ensure that the proper amount of time has passed before sending the row command of the current access request to the memory 802. Once the proper amount of time has passed, the controller 804 sends (1212) the current access request (comprising a row command and a subsequent column command) to the memory

802. The controller 804 loops back to receive (1202) another request from the external component 806.

Returning to (1208), if the controller 804 determines that the current access request is not directed to the same memory portion 812 as the immediately preceding access

5 request, then it proceeds to determine (1214) the proper timing for sending the current access request relative to the previous access request. In doing so, the controller 804 (when possible) times the sending of the current access request such that it causes the memory 802 to output data onto the data bus 820 (in the case of a read operation) with substantially no idle time between data sets. By doing so, the controller 804 enables the memory 802 to

10 achieve full data bus utilization. In some cases, it may not be possible for the controller 804 to time the requests such that full data bus utilization is achieved. For example, when there is a large time interval between requests from the external component 806, it may not be possible for the controller 804 to send access requests which are close enough in time to achieve full data bus utilization. In such cases, the controller 804 sends the current access

15 request immediately to the memory 802 to minimize idle time on the data bus.

A point to note is that since the current access request is directed to a different memory portion than the immediately preceding access request, the controller 804 does not need to impose any minimum time delay between the sending of the access requests. Thus, if so desired, the row command of the current access request may be sent immediately after

20 the row command of the preceding access request. In an SDRAM, the row command of the preceding access request and the row command of the current access request may be sent in consecutive clock cycles. Once the proper timing is determined, the controller 804 sends (1212) the current access request to the memory 802 at the proper time to cause the memory

802 to be accessed. The controller 804 loops back to receive (1202) another request from the external component 806. In the manner described, the controller 804 interleaves access requests between the memory portions 812 whenever possible to enable full data bus utilization. At the same time, it imposes time delays when necessary to prevent data errors.

5

At this point, it should be noted that although the invention has been described with reference to a specific embodiment, it should not be construed to be so limited. Various modifications may be made by those of ordinary skill in the art with the benefit of this disclosure without departing from the spirit of the invention. Thus, the invention should not be limited by the specific embodiments used to illustrate it but only by the scope of the appended claims.

10